Identifying and classifying promoter DNA sequences in the bacterium *Escherichia coli*

Miria Rafante Bernardino

Supervisor: Prof. Dr. Robert Beiko

Dalhousie University

This report is submitted for the Research Aptitude Defense

September 2020

Abstract

Genes encode proteins that are responsible for many tasks including cell growth and heat shock response. Gene expression, the first step in the production of proteins, is regulated to ensure that the cell has the appropriate response according to its current needs. Promoter sequences are special segments in genomic DNA that regulate the expression of genes. Automated identification of promoters can help to predict when genes are likely to be active, and which sets of genes are likely to be expressed at the same time. However, their short length and high variability make them difficult to distinguish from non-promoter DNA sequences.

Several tools have been developed to identify promoters in DNA. For instance, iPromoter-2L and CNNProm are two state-of-the-art approaches to identify promoters and reported a sensitivity of 79.2% and 90%, and specificity of 84.16% and 96%, respectively. However, these tools' tests and validations commonly stop with small fixed length sequences despite the real-life problems are genome-scale problems. Consequently, identifying promoters in an *Escherichia coli* genome with such tools remains a challenge. This chromosome is comprised of a single DNA sequence with more than 4.6M nucleotides and around 3,000 promoters. Therefore, even a tool with a high specificity of 90% will predict around 460,000 false-positive promoters. In this work, we developed PromClass that identifies promoters on the genome and aims to minimize false positives. We used PromClass and two state-of-the-art tools to identify the promoters present in a segment of *E. coli* genomic sequence. The results show that PromClass finds the highest number of promoters (61% against 48%) while keeping the lowest number of false-positives (7% less than the model with the smallest number of false-positives).

Table of contents

List of figures

List of tables

Acronyms

AUC area under the curve. 13

ML machine-learning. 6

MLP multi-layer perceptron. 11

NCBI National Center for Biotechnology Information. 8, 12

NN neural network. 6

PseKNC pseudo k-tuple nucleotide composition. 5, 8

PWM position weight matrix. 5

RF random forest. 6, 9, 11, 17

SSP sequence starting point. 12

TBS transcription binding sites. vii, 2, 4

TSS transcription starting site. vii, 4

Introduction

1.1 Introduction

Promoter sequences are short DNA segments that play a key role in controlling the expression of genes (Hohmann-Marriott and Lale (2020)). Promoters fall into several classes, with each class responsible for expressing genes at different times in the bacterial cell's growth and development, and in accordance with its survival needs (Lin *et al.* (2017), Testerman *et al.* (2002), Davis *et al.* (2016), Liu *et al.* (2018)). Finding and characterizing these regions is fundamental to understanding how genes function and are regulated (Hohmann-Marriott and Lale (2020)). The classic and reliable method to identify or confirm the existence of promoters is through laboratory experiments that are based on finding physical and or genetic evidences. (Weiss *et al.* (2013)) Unfortunately, these experiments are demanding, time-consuming and costly (Weiss *et al.* (2013), Liu *et al.* (2018), Silva and Echeverrigaray (2012), Lin *et al.* (2014)). Yet, RegulonDB (Gama-Castro *et al.* (2016)) is a platform that offers experimentally confirmed *Escherichia coli* promoter sequences that were mined from several papers and made it accessible. Fig. 1.1 is an example of how the information is available.

An alternative approach to experimental assessment is to develop computational models that predict promoters with high accuracy. However, promoter prediction is a challenging problem: their short length and high variability make them difficult to distinguish from non-promoter DNA sequences (Hohmann-Marriott and Lale (2020)). Additionally, there is a dearth of large high-quality labelled training sets, especially outside of the best-studied organisms such as humans and the bacterium *E.coli* (Gama-Castro *et al.* (2016)).

To date, predictive methods have typically suffered from high false-positive rates and low accuracy. For instance, an approach with 90% specificity when applied to *E. coli* genome that has approximately 4.6M nucleotides, which with a sliding window of 81 nucleotides, has more than 4.6M promoter candidates, may predict 460,000 promoters that do not exist.

Therefore, even where methods have been developed, they have limited utility as they are not fit for genome-scale applications (Silva and Echeverrigaray (2012)).

There is no universal promoter structure. There are many differences between the promoters of prokaryotes, which are usually single-celled and lack a nucleus, and eukaryotes, which can be multicellular and have cells with nuclei. The "machinery" of gene expression differs between the two groups, and the DNA structures related to the transcription process (Kornberg (1999), Shafee and Lowe (2017)) are different as well. Although the promoters of prokaryotes such as *E. coli* tend to be simpler than those of eukaryotes, the key feature-encoding and pattern-recognition challenges are common to both. In this study, our focus is on prokaryotes.

In the cell, promoters are recognized and bound by *RNA polymerase* (RNAP), a group of proteins with a central role in transcription, where the DNA sequence of a gene is used to produce a complementary RNA molecule that is used as the template to produce a protein. The gene can be on either of the two DNA strands. These strands are kept together due to hydrogen bonds that occur between nucleotides A and T, and nucleotides G and C, resulting in the base-pairs AT, TA, GC or CG. One of these strands is conventionally chosen to be the forward strand and has the first base of the base-pair. The other one is the reverse strand, also called reverse complement, and has the second base of the base-pair. DNA binding has a bigger contribution from one of the strands (Paget (2015)) and is mediated by one protein in the RNAP complex, the σ factor(Davis *et al.* (2016)). There are multiple σ factors in a cell, and specific classes of promoters are recognized by specific σ factors. In *E. coli*, there are seven known sigma factors: σ19, σ24, σ28, σ32, σ38, σ54 and σ70. Different σ factors are responsible for expressing different sets of genes: for instance, σ24 and σ32 participate in the expression of genes involved in heat-shock response. By contrast, σ70 expresses a large number of "housekeeping" genes that are expressed frequently or continuously (Silva and Echeverrigaray (2012), Davis *et al.* (2016)). A promoter can be identified by a single type of σ factor only, hence the promoter types refers to the type of the σ factor that binds them (Liu *et al.* (2018)).

After identifying the promoter, the σ factor transcription binding sites (TBS) inside the promoter are used to initiate transcription. For most promoters, the TBS are a pair of two small sets of nucleotides separated by a set of more-variable nucleotides (Davis *et al.* (2016), Gama-Castro *et al.* (2016)). For the purpose of this study, nucleotide and base-pair (bp) have the same meaning. The overall number of nucleotides composing the binding sites and spacer (sequence between the TBS) in *E. coli* is not greater than 58 bp according to the samples at RegulonDB. In addition to their small size, promoters are not strictly nucleotide-composition conserved sequence or structural conserved sequences. For instance, their composition is

```
# Columns:
# (1) Promoter identifier assigned by RegulonDB
# (2) Promoter Name
# (3) DNA strand where the promoter is located
# (4) Genome map position of Transcription Start Site (+1)
# (5) Sigma Factor that recognize the promoter
# (6) Promoter Sequence (+1 upper case)
# (7) Evidence that supports the existence of the promoter
# (8) Evidence confidence level (Confirmed, Strong, Weak)
ECK125230824    TSS_10      forward 38      unknown aaacgccttagtaagtatttttcagcttttcattctgactgcaacgggcaatatgtctctGtgtggattaaaaaaagagtg  [...]  Strong
ECK125230832    TSS_10      forward 2581    unknown tttgccgcgcgcgtggcgaaggcccgtgatgaaggaaaagttttgcgctatgttggcaatattgatgaagatggcgtctgc  [...]  Strong
ECK125230912    TSS_100     reverse 55146   unknown gtagcgaccagtaactccagcattgaataccgtcgggatgaagaccgtctggtacagctgAattaccgttacgccagcccg  [...]  Strong
ECK125236475    TSS_1000    forward 892926  unknown ttagtcgcaatcacattactgatcatggttttgcctgcgctttttgcgtaagctgtgccgGtctttttatcgaaagaggtt  [...]  Strong
ECK125236476    TSS_1001    reverse 900345  unknown aaatgcaggcagaatgtacttttactaatcacgcgttcgacagcctgatcccgtccctgaAattcagaaaaatatgacgccg  [...]  Strong
ECK125236477    TSS_1002    reverse 900605  unknown gtttattgcatataaattcacttgatggcattgttatcccatgccgcagacacggccaaaAatcataagattgacagacgg  [...]  Strong
ECK125236478    TSS_1004    reverse 902050  unknown gttcagcgctggtaaccattctgcgtggcctgccagaaattctggtggtgctgtttatctAttttggctcctcgcagctgc  [...]  Strong
ECK125236479    TSS_1005    reverse 902184  unknown tgaatttttttcctttagcaagcgccgccgggatgaccgtcggccttgccgtttgtgcattGattgtcgggctggcgctggc  [...]  Strong
ECK125236480    TSS_1006    reverse 902897  unknown gcaggttttagtctttccgccacagctgccgaaaccattcgttttgctaccgaagcctccTatcctccgtttgaatcgatt  [...]  Strong
ECK125236481    TSS_1007    reverse 903004  unknown gacgcgagctgctttaccgagccgcaaaccgaagcatttaaaaactatctctctcactaaGattcgggataacaacaatga  [...]  Strong
ECK125236482    TSS_1008    reverse 903017  unknown cgtagaacaaggcgacgcgagctgctttaccgagccgcaaaccgaagcatttaaaaactaTctctctcactaagattcggg  [...]  Strong
```

Fig. 1.1 Segment of one of the files available in RegulonDB with promoters. In general, the first lines explain the meaning of each column the data is organized. In this example, the first entries of this file are promoters with sigma factor not yet determined (column (5) Sigma Factor that recognize the promoter).

Table 1.1 Number of samples from each sigma type available on RegulonDB-Release 9.4 with *Evidence confidence level* "strong" or "confirmed"

Sub-class	Number of samples
σ24	65
σ28	9
σ32	58
σ38	103
σ54	17
σ70	753

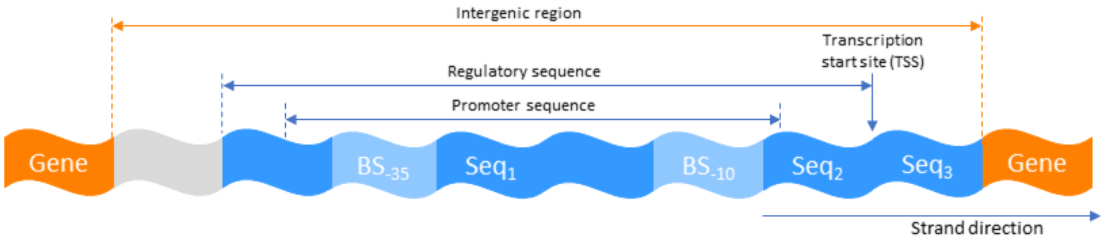

Fig. 1.2 Details showing a standard intergenic DNA region that has a promoter sequence. In this example, this promoter sequence is associated with the gene whose starting point is immediately after the end of the promoter sequence. The image shows important intergenic components and how they are related to a promoter sequence. TBS_{-35} is the binding site that is further from its gene. TBS_{-10} is the binding site closer to the gene. *Seq* are segment of sequences (1) between the binding sites pair, (2) between TBS_{-10} and the transcription starting site (TSS), and (3) between the TSS and the gene.

diverse (Hohmann-Marriott and Lale (2020), Silva and Echeverrigaray (2012)), they may not have a pair of TBS but a single one (Paget (2015)). In the samples at RegulonDB, there is an assortment of lengths for TBS, for the sequence between a pair of TBS and between a gene and its TBS. All these mixtures make it hard to differentiate promoters from non-promoters, and even between promoter types (Hohmann-Marriott and Lale (2020)). This is a possible reason why promoters are not as well annotated and understood as most genes (Silva and Echeverrigaray (2012)).

1.2 Encoding promoters

Current classification methods typically make use of several different approaches to encoding DNA sequence information. One popular approach is to count the frequency of k-mers, an enumeration of all "words" of length k in a sequence of DNA. Compositional variation in promoter sequences may be reflected in significant over-representation or under-representation of specific words. For instance, for k=2 there are $4^k = 4^2 = 16$ possible words: AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC, TG, TT. With a bigger value of k, the relative ordering of the nucleotide information captured by k will be longer but the feature vector will be bigger and more sparse. Despite keeping the ordering of each consecutive k nucleotides, k-mers do not consider the relative ordering of the corresponding words in the DNA sequence. Other methods, however, are based on the frequencies of nucleotides of DNA at each position in a set of promoters. These frequencies can be summarized in

a structure such as a position weight matrix (PWM) (Gordon *et al.* (2003), Hooghe *et al.* (2012)), which can in turn be used to score unknown sequences.

In addition to sequence information, structural features of DNA can be useful in classification. The DNA double helix is not regular, and its shape and stability are influenced by the sequence of nucleotides. These properties have been summarized in feature sets such as pseudo k-tuple nucleotide composition (PseKNC) (Chen *et al.* (2014)), which decomposes the sequence into k-mers and maps them to physicochemical property values specific for each word that is used to calculate scores. Each score (Θ_i) is concatenated to the k-mers decomposition and refers to all k-mers i nucleotides distant from each other in the sequence. Eqs. 1.1 and 1.2 show how Θ_i is calculated:

$$\Theta_i = \frac{1}{n-k+1-i} \sum_{j=1}^{j=n-k+1-i} \theta_j \tag{1.1}$$

$$\theta_j = \frac{1}{p} \sum_{\rho=1}^{\rho=p} [prop_\rho(kmer_j) - prop_\rho(kmer_{j+i})]^2 \tag{1.2}$$

Where p = number of properties, $Prop_\rho$ access the list of k-mer for property ρ, n is the sequence length, $kmer_j$ is the j-th k-mer in the sequence and i $\in$ {1, 2, ..., λ} (Chen *et al.* (2014)). λ is the furthest distance between the k-mers one wants to calculate, the further the two k-mers are in the sequence, the longer-range information is extracted (Chen *et al.* (2014), Dao *et al.* (2019)). Chen *et al.* (2015) recommend the use of $\lambda = 1$ or $\lambda = 2$ to avoid over fitting. Liu *et al.* (2018) used PseKNC with $\lambda = 2$ to build the model called iPromoter-2L.

The last approach we want to mention neither brings outside information about the sequence nor represents the sequence by representing its parts independently. Instead, the approach encodes a sequence by hot-encoding its nucleotides into a 2D array, i.e., replacing nucleotide A by [1, 0, 0, 0], T by [0, 1, 0, 0], G by [0, 0, 1, 0] and C by [0, 0, 0, 1]. Umarov and Solovyev (2017) used this strategy when building their model called CNNProm.

1.3 Prior work

The PWM approach counts the frequency of the nucleotides in each position of the promoter sequence and uses it to find the probability of a query to be a promoter. This approach favors most common promoters. It can have many false-positive matches as a result of the small length of promoter sequences and can have many false-negatives due to the poor conservation of promoters (Hohmann-Marriott and Lale (2020)). Therefore, machine-learning techniques

emerge as a possible solution. machine-learning (ML) methods are able to model patterns in more sophisticated ways potentially increasing the accuracy of predictions made on previously unseen instances (Min *et al.* (2016), Pedregosa *et al.* (2011)). Many promoter-recognition approaches have been developed using several ML techniques.

Demeler and Zhou (1991) used neural network (NN) to train their solution on eighty promoter sequences and non-promoter samples composed of random nucleotides. Since random sequences can be quite different from promoter sequences, it may make the classification problem easier to solve and caused their claimed accuracy to be as high at 100%. Later, Alipanahi *et al.* (2015) developed DeepBind also using NN. DeepBind has a collection of 927 "motif detector" models representing transcription factors and RNA binding proteins. The tool receives a sequence with 14 to 101 nucleotides, scans it using the motif detectors and rectifies it by identifying positions with a good pattern match and bringing its negative values to zero. Then it computes the maximum and average values for each motif detector's rectified response before feeding these scores to a NN. The positive samples came from formerly published transcription factors and the negative set was built by shuffling the positive sequences matching the dinucleotide composition (i.e., k-mers with k = 2). Other approaches have been applied to promoter detection including genetic algorithms and evolutionary computing (Beiko (2003), Beiko and Charlebois (2005)), followed by adaptive quality-based clustering (AQBC) (Florquin *et al.* (2005)), hidden Markov models (Mallios *et al.* (2009)), Mahalanobis discriminant (Lin and Li (2011)), partial least squares classifiers (PLS) (Song (2012)), support vector machines (SVM) (Xiao *et al.* (2018), Zuo and Li (2010), Lin *et al.* (2014), Lin *et al.* (2017), Lai *et al.* (2019)), random forest (RF) Liu *et al.* (2018), and convolution neural network (CNN) (Umarov and Solovyev (2017), Tayara *et al.* (2020)).

Several recent approaches have yielded promising results. iPro70-PseZNC is the solution created by Lin *et al.* (2017) using SVM mentioned above. In their solution they introduced a modified version of the Z-curve method (described below) called multi-window Z-curve and combined it with PseKNC for dinucleotides, creating the *pseudo multi-window Z-curve nucleotide composition* or *PseZNC*. The Z-curve method is used to obtain sequence features by calculating the nucleotide frequencies and mapping them into a 3D space in which each axis is a linear combination of all individual nucleotide frequencies. The multi-window Z-curve, however, can represent the frequencies of the nucleotides from a non-fixed length sequence. Distinctly, their negative set was created by randomly selecting sequences from genome coding and non-coding regions whereas the positive samples came from $\sigma 70$ promoter samples available at RegulonDB. Recently, the use of RegulonDB as the source of promoter samples became popular. CNNProm is the approach developed by Umarov and Solovyev (2017) using CNN. They also used the $\sigma 70$ promoter samples from RegulonDB

but the non-promoter samples were extracted from the reverse complement of randomly selected genes. The sequences were encoded using a one-hot encoding for each nucleotide, i.e., (1, 0, 0, 0) for A, (0, 1, 0, 0) for T, (0, 0, 1, 0) for G and (0, 0, 0, 1) for C, and inputted into a 2D convolution that feeds a fully connected layer. They reported a sensitivity of 90% with a specificity of 96% on an *E.coli* test set. Liu *et al.* (2018), on the contrary, chose RF. Their solution, called iPromoter-2L, first classifies the queries into promoters and non-promoters and then classifies the promoters further into σ-factors, i.e., σ24, σ28, σ32, σ38, σ54 or σ70. The promoter samples were obtained from RegulonDB, then they were compared two by two and discarded if their nucleotide composition is at least 80% identical. The non-promoters were extracted from long coding regions and convergent intergenic regions, i.e., intergenic regions that are not upstream of any of the genes flanking it (Lin *et al.* (2014)). Their reported metrics for the binary problem is 79.2% of sensitivity and 84.16% of specificity. iPromoter-2L (Liu *et al.* (2018)) and CNNProm (Umarov and Solovyev (2017)) are the baseline for this study.

After we began this project, several techniques were released. These more recent approaches feed their training and testing sets to different classifiers to then choose the classifier that will compose their solution. Rahman *et al.* (2019) developed iPro70-FMWin by comparing SVM, logistic regression (LR), k-nearest neighbor (KNN), decision tree, gaussian naive Bayes (GNB) and linear discriminant analysis (LDA). They say SVM showed the best performance. Tahir *et al.* (2020) developed 2L-iPSW(word2vec) by testing KNN, SVM and CNN with CNN outperform the other two. Tahir *et al.* (2020) use natural language processing (NLP) technique to encode DNA segments.

Objective

The purpose of this project is to develop a classifier to identify promoters in genomic DNA with an emphasis on low false positive rates. In addition to the widely used k-mer strategies, we exploit physicochemical properties to reflect the structural nature of protein-DNA interactions. To control the false-positive rate we add a second layer that merges overlapping predictions to the basic classifier. into a final prediction. Finally, to evaluate our

results, we compare the performance of our approach on a large segment of the genome to the performances of two state-of-the-art methods.

Methods

3.1 Data

3.1.1 Promoter and non-promoter samples

The positive data set comprised of the promoters available on RegulonDB-Release: 9.4 from 05-08-2017, with a confidence level of "strong" or "confirmed". Each promoter sequence is labeled according to its σ factor and the number of samples per class is unbalanced as shown in Table 1.1. The table also shows the six σ factors we have samples for from the seven σ factors found in *E. coli*: σ19, σ24, σ28, σ32, σ38, σ54 and σ70. These samples are referred to as positive promoters or σ++. There are 1005 of these samples.

The negative set was created by first aligning the promoter sequences available at RegulonDB to an *E. coli* K12 MG1655 genome from National Center for Biotechnology Information (NCBI) (NCBI_Assembly:GCF_000005845.2). This alignment creates the ground truth for promoters in the *E. coli* genome. All segments in the DNA with more than 58 nucleotides and without a match to a promoter sequence were considered for the negative data set, i.e., intergenic and coding regions. Then, we used a sliding window with 58 bp to create sequences of fixed length, but to have a balanced number of samples among the classes, we selected 1008 samples that were equally distant from each other along the genome. It is possible though that a negative sample is actually a promoter sequence yet to be discovered.

3.1.2 Feature extraction

Four different feature sets were extracted in this work.

- (1) *pseknc* - To extract the *pseknc* features from the segments we implemented the PseKNC algorithm described in Chen *et al.* (2015) with some changes to be more efficient by being trinucleotides and DNA focused. The algorithm decomposes the

sequences into k-mers with k=3 and calculates Θ_1 and Θ_2 as shown in Eq. 1.1. Θ_1 and Θ_2 are two numbers that summarizes the physicochemical properties of the entire sequence by accounting the properties for all pairs of 3-mers one and two nucleotides a part, respectively. The table that maps the 3-mers to their properties was extracted from Chen *et al.* (2015)'s work. The final set is composed by 3-mers frequency followed by Θ_1 and Θ_2.

- (2) *several k-mers* - Calculates the frequency of each k-mers for $k \in \mathbb{Z} | 1 \leq k \leq 6$ in the sequence. The bigger the k, the sparser the corresponding array will be.
- (3) *physicochemical per 3-mers* - Extends *pseknc* by preserving the 3-mers position in the sequence and then mapping each 3-mers into its individual 12 physicochemical properties (obtained from Chen *et al.* (2015)). A sequence with fifty eight nucleotides has fifty six 3-mers which generates 672 *physicochemical per 3-mers* features.
- (4) *physicochemical stats* - It is the statistical measures of minimum, maximum, mean, standard deviation, median and variation over the *physicochemical per 3-mers* features described above.

The negative and positive samples from Sec. 3.1 were encoded using the four approaches above and the pairs: (1) with (3), and (2) with (3). The code to generate features (1) to (4) is available at https://beiko-lab.github.io/DNA_encoders/.

3.2 Classifiers

3.2.1 Binary Random Forest

RF is the classifier chosen in our first attempt to identify and classify promoters. RF is composed of decision trees that are built with random features chosen from the feature pool. Each tree votes for a class for a given input x. RF then returns the most popular class among the collection of votes. The high dimensionality of the encodings listed above, and the disparity observed in the t-tests shown in Fig. 3.1 suggested the use of a decision-based process. The implementation used here is the one at *sklearn.ensemble.RandomForestClassifier* (Pedregosa *et al.* (2011)). Table 3.1 shows the scope of the parameter search.

The RF classifier was trained and tested using each of the six feature sets from Sec. 3.1.2 on the 1008 samples from σ- - and 1003 samples from σ++. We used a 5-fold cross-validation and stratified splitting to slice the data into training and testing sets keeping the proportion of σ++ and σ- - constant. At each split, four parts were used for training while

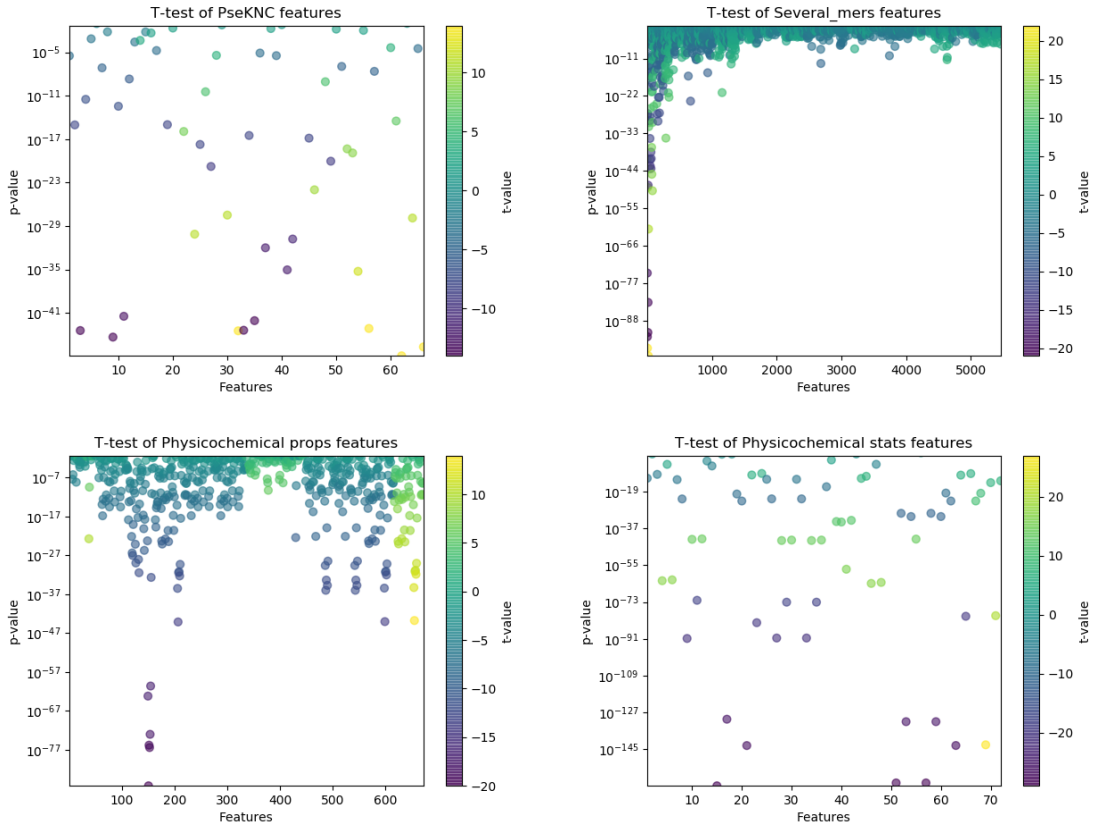

Fig. 3.1 T-test applied to each of the features from the four data sets in Sec. 3.1.2. Each point on the plots represent a feature on axis x. Their p-value are on the axis y and the t-value is shown on their color mapped to the side bar of each plot. All the points in the plots have a p-value $\leq 10^{-3}$, i.e., they refer to features that differ between promoters and non-promoters sets with a confidence level of more than 99%. t-values measure the ratio of samples' distances between classes by their distances inside the classes. The positive and negative signals indicates the direction of the difference between the means of the two sets. As we do not have interest to know what set has the highest mean for each feature, we will use the t-value absolute number. The greater the t-value is, the more similar the features are inside of the classes them between the classes.

Table 3.1 Parameters used with the Random Forest Classifier implementation in sklearn package

Parameter	Value
numbers of trees	10, 50, 100
function to measure the quality of a split	"*entropy*"
number of jobs in parallel	-1 (i.e., using all processors available)
random state	2019
warm start	True
classes weight	None

one part was used for testing. For each set of features, three models were created, with a hundred, fifty and ten trees.

3.2.2 Multi-layer perceptron

The RF results (Table 4.1) show a high false-positive rate suggesting that rule-based approaches are not able to satisfactorily uncover the promoters' patterns, but an approach able to deal with latent-features by using a more complex and robust structure such as a multi-layer perceptron (MLP) might be more successful. The RF results also were better for the feature extraction set with the *physicochemical per 3-mers* encoding. Therefore, we decided to use this set together with *several k-mers* that is the only other set that is not contained or can be obtained from *physicochemical per 3-mers* to develop the "PromClass" method using a paired MLP approach. The MLP models use Keras (Chollet *et al.* (2015)) deep learning framework. To establish the number of hidden layers, hidden units and what feature extraction to use, we used a 10-fold cross-validation with 10% of the training set reserved for validation and a grid search. The search space for number of hidden layers was between 1 and 5, for the hidden units was between 12 and 128, and for number of epochs was between 5 and 500. Among the models tested, we chose one with the highest accuracy on the validation set and named it "L1". As shown in the diagram of Fig. 3.3-(a), L1 is the model used in the first classification layer of PromClass. We used three fully connected hidden layers with 128 hidden units each, all of them with L2 Regularization, i.e. the variables' coefficients are penalized by adding their squared value to the loss function at the rate of 1% in order to minimize the impact of over-fitting. The model trained for 100 epochs, used the samples one single time on each epoch, had a learning rate of 0.01, used ReLU as the activation function for all layers except the last one that used a Sigmoid, and used Adam as the optimizer.

L1 is a binary classifier, i.e., it was trained to classify a query into promoter (σ ++) or non-promoter (σ –). However, as mentioned before, promoters have types (classes) named according to the σ-factor that binds them and, although focusing on the binary problem, the following models were built as an assessment on the difficulty of classifying promoter types:

- Seven-class model classifying for the classes σ – and σ 24, 28, 32, 38, 54 and 70
- Six-class model classifying for the classes σ 24, 28, 32, 38, 54 and 70

The multi-class classifiers used the structure and hyperparameters from L1 with a few extensions. The seven-class model was trained for 150 epochs, had a learning rate of 0.001 and used classes weights to force all classes to have the same importance. We used the same 1008 samples for σ – as for L1 and all the samples from each σ factor. The loss function also changed from a binary cross-entropy to a categorical cross-entropy. The six-class model used the same parameters as the seven-class model except the classes and classes' weights that were updated for the six-class problem. This model assumes the query is a promoter and assigns it to one of the six promoter types.

3.3 Predicting promoters

The L1 model receives DNA sequences with 58 nucleotides encoded using *several k-mers* and *physicochemical per 3-mers*, and gives the probability of each sequence to be a promoter. However, the *E.coli* genome, for instance, has more than 4 million nucleotides. Finding the promoters in the genome is similar to a natural language processing problem, however there is no natural spacing between the "words" so we do not know where each word starts or ends. To deal with this challenge, we pre-process the DNA as shown in Fig. 3.3-(a). The DNA is the genome of *E. coli* K-12 MG1655 from NCBI platform at https://www.ncbi.nlm.nih.gov/. NCBI provides access to genomes, genome's annotations, proteins, documents, research, and several other sources of information on genomics. To pre-process the DNA, first, we assume that there is a window starting on all nucleotides, i.e., each nucleotide will be the word starting point or sequence starting point (SSP) - as in the diagram - for a given word. To be consistent with L1 training and testing, we encoded the genome in overlapping windows 58 nucleotides in length.

The first 27177 DNA positions were used in our previous analysis and therefore were put aside on this phase. Thus, the pre-process started to be applied on position 27178. However, due a time constraint we had to stop this execution that had time to pre-process only up to position 158018. The encoded sequences of fixed length are encoded as input

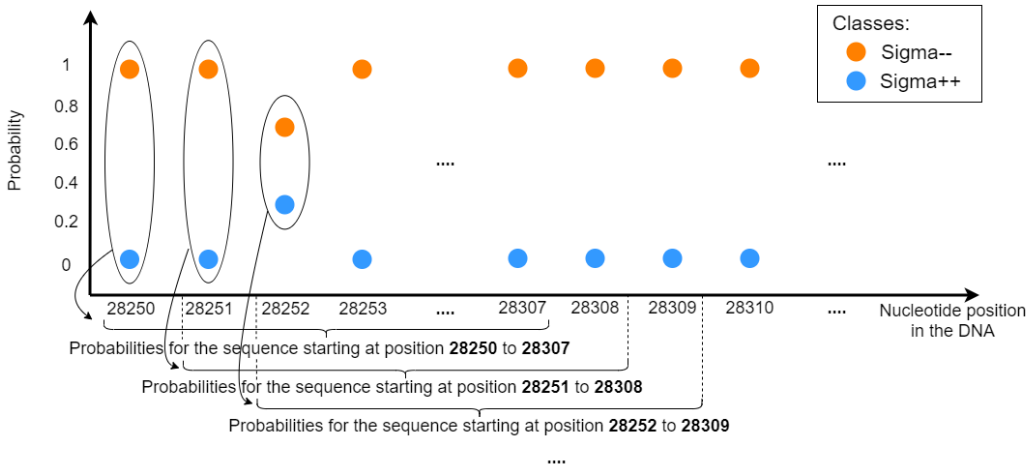

Fig. 3.2 Diagram explaining the probabilities output by the first layer of (two-class) classification. Each orange dot with coordinates (x, y) is the probability y between 0 and 1 of the sequence with 58 nucleotides starting at position x to be a *Sigma- -*. Accordingly, each blue dot with coordinates (x, y) is the probability y of the sequence with 58 nucleotides starting at position x to be a *Sigma++*.

to L1 that attributes a probability to each 58-nucleotide window to be a promoter. Yet, the sequences overlap each other as shown in the diagram of Fig. 3.2. Since each nucleotide is included in 58 separate overlapping windows, it will have 58 associated probabilities to be a promoter. To exploit the information associated with each nucleotide, we built a second classification layer for PromClass, called L2. The diagram in Fig. 3.3-(b) shows how we organized the predictions of close positions and built the feature vectors. The feature vectors were composed by the probabilities of the sequences that contain a given nucleotide, on the forward and reverse strands. These vectors are 116 long, encompassing 58 probabilities for the sequences on the forward strand and 58 probabilities for the sequences in the reverse complement. Then, we set the feature vector label to one if the corresponding nucleotide sequence overlapped with a promoter mapped from RegulonDB (Sec. 3.1.1) on its forward or reverse strands, and to zero otherwise. This is because in most cases segments of both strands contributes with hydrogen bonds to bind the sigma factor (Lin and Guo (2019)). At this stage, we had 129,907 *Class 0* and 934 *Class 1* samples to train and test L2.

We modified the approach in L1 to build the L2 model. For L2, we explored the hyper-parameters used in the grid search for L1 and used class weights given the very unbalanced number of samples per class. We also calculate the area under the curve (AUC) for each model in order to select the best ones. However, the highest AUC of 83% was produced by two models, L2-A and L2-B. Both models were trained for only 60 epochs, reaching a

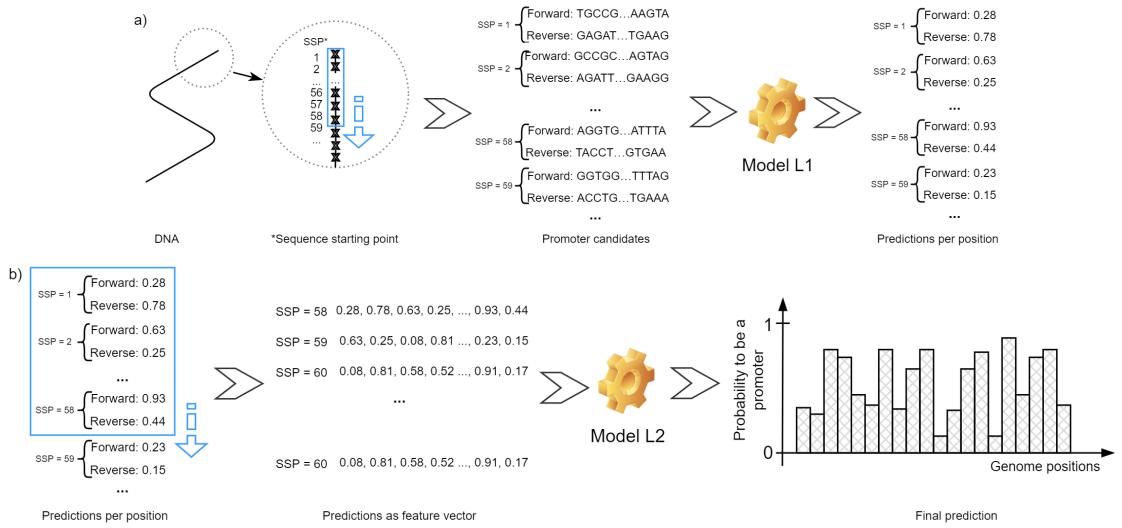

Fig. 3.3 The PromClass workflow. First, (a) Model L1 is applied to the promoter candidates that are sequences extracted from the DNA using 58bp sliding windows (indicated by blue arrows). Then, (b) Model L2 is applied to L1 predictions organized into a feature vector. Its output is filtered into promoter predictions.

plateau before the 60^{th} epoch, L2-A classes' weights were leveraged by 10 and L2-B's by 5. These models make predictions for each nucleotide. Therefore, it needs to be properly combined into promoters' predictions. Thus, we developed a method (Fig. 3.4) to filter predictions based on the number of nucleotides in a given window that were flagged by L2 as promoters. The filter is a simple function that, given a window size, counts the number of nucleotides k whose prediction is above a threshold t. If k is bigger than a threshold ρ, we add that window to the list of predicted promoters. In our case, L2-A and L2-B were filtered using the same sequence length used by CNNProm and iPromoter-2L, i.e., l=81bp. The parameter ρ of our filter was chosen based on the lengths of the segment between and including the binding sites for the samples from RegulonDB. The mean length is 28.8 with a std of 3.7. Because we do not want a high number of false positives, which is our focus as explained before, we are considering only one standard deviation and set ρ=25, i.e., the minimum number of nucleotides above the threshold t needs to be at least 26 so the segment can be considered a promoter.

Finally, we compared the predicted promoters with the ground truth mapped into the genome. A positive prediction for promoter outputs the location where the promoter was found on the genome. In our evaluation, a prediction is correct, i.e. it is a TP, if the yield location overlaps the position of a true promoter by at least one nucleotide. To add one to the TP count, a prediction needs to overlap a true promoter that was not yet overlapped by another

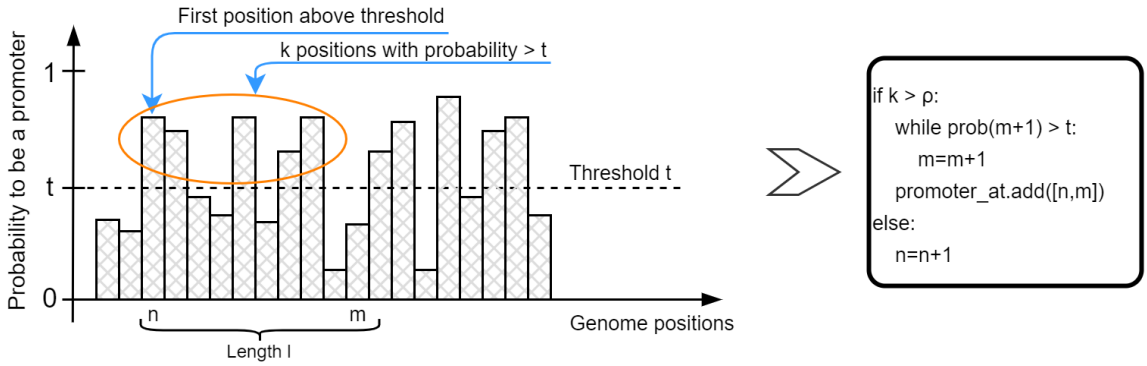

Fig. 3.4 Filtering predictions with model L2. The threshold t determines what nucleotides may be in a promoter, but it is the filter that in fact checks what can be a promoter. The parameters l and ρ specify a filter. In this study, l = 81 and ρ=25. The filter's output is a list with the location of each promoter.

prediction. If it does not overlap, it is a FP. If no promoter prediction overlaps a true promoter, it is a FN. Similarly, a TN should be the sequences overlapping true non-promoter regions and predicted as non-promoter. However, non-promoter regions are commonly far greater than 81bp that is the sequence length used for most of the promoter prediction solutions. For instance, consider a non-promoter sequence A with 3,000 nucleotides. It is not reasonable to have more TN than N samples, therefore only one prediction for non-promoter associated with sequence A could be accounted as TN. However, it is possible that doing this we could account the entire sequence A as correctly predicted when the prediction actually refers to only 81 of its nucleotides. Because it does not look correct, we decided we should not account for TN, but only for TP, FP and FN values. With these three values one can calculate sensitivity and precision (see Sec.3.4), and evaluate the model's performance on the genome.

We also applied the solutions iPromoter-2L and CNNProm to the same segment of the genome. For CNNProm, we were able to submit the entire segment wrapping it into 81bp on a single fasta file. Their solution to predict promoter on sequences with different lengths is available at http://www.softberry.com/berry.phtml?topic=cnnpromoter_b&group=programs&subgroup=deeplearn. For iPromoter-2L we created a fasta file for each 5 thousand nucleotides and submit them one by one to http://bioinformatics.hitsz.edu.cn/iPromoter-2L/server. The sequences in the files were overlapping each other by 100bp to guarantee the entire segmented would be evaluated by the solution. Then, we accounted for their TP, FP and FN considering only one single prediction for each sequence and compared their results with our solution using a precision-recall (sensitivity) curve. This curve is more appropriate to measure performance of models on unbalanced databases than the ROC curve (Saito and Rehmsmeier

(2015)). CNNProm output is a list of the promoters predicted with a position number, for example, *Promoter Pos: 15138 Score- 0.999*. As we did not find information on where the predicted promoter is with respect to the position *Pos* informed in the list of predictions, we account for the TP, FP and FN assuming two cases. First, the predicted promoter ends at position *Pos*. Second, the predicted promoter starts at position *Pos*. We kept the case with the best result: highest TP and lowest FP and FN. Regarding iPromoter-2L, this solution uses a sliding window with 81bp on the genome-query and outputs one prediction for each sequence. This query has 130,841 nucleotides and iPromoter-2L evaluated over 130 thousand sequences and output the same quantity of predictions. For each promoter prediction we checked if there was an overlap with a true promoter and account all the true promoters with at least one prediction overlapping it as the TP. All the promoter predictions that did not overlap a true promoter were accounted as FP.

3.4 Evaluation metrics

We used sensitivity, specificity, precision, and accuracy performance metrics to measure how well our models can generalize making accurate predictions for unseen data as well as to compare our approach to previous work's prediction power on real-life problems. These metrics are defined in Eqs. 3.1, 3.2, 3.3 and 3.4 (Liu *et al.* (2018), Saito and Rehmsmeier (2015), Umarov and Solovyev (2017)), respectively. Acronyms in the equations are explained below.

- FN(c) = number of elements $\in$ class c and classified in a class other than c
- TP(c) = number of elements $\in$ class c and classified as c
- P(c) = number of elements $\in$ class c
- FP(c) = number of elements $\notin$ class c and classified as c
- TN(c) = number of elements $\notin$ class c and classified in a class other than c
- N(c) = number of elements $\notin$ class c

$$sensitivity(c) = \frac{TP(c)}{P(c)} \tag{3.1}$$

$$specificity(c) = \frac{TN(c)}{N(c)} \tag{3.2}$$

$$precision(c) = \frac{TP(c)}{TP(c)+FP(c)} \tag{3.3}$$

$$accuracy(c) = \frac{TP(c)+TN(c)}{P(c)+N(c)} \tag{3.4}$$

Results

4.1 Binary classifiers

We selected the RF model with the highest accuracy trained on each of our six feature sets. The feature set used in each model, the parameters as well as their accuracy are shown in Table 4.1. The RF models using feature extraction (1), (2) and (4) had performance very similar around 78%. The models using the features from (3) and the combinations (1) and (3), and (2) and (3) had higher accuracy around 83%.

Several models were trained using MLP to be in the PromClass first classification layer (L1) (Fig. 3.3-(a)). Their performance was similar around 85% of accuracy on the validation set. The chosen model, however, presented validation accuracy around 89%. Fig. 4.1 shows the accuracy and loss on the train and validation sets at the end of each epoch during training

Table 4.1 Binary models' parameters and performance.

Classifier	Features set	# of trees	Accuracy	Sensitivity	Specificity
RF	*pseknc* (1)	50	0.77	0.77	0.78
RF	*several k-mers* (2)	100	0.79	0.79	0.79
RF	*physicochemical per 3-mers* (3)	100	0.83	0.83	0.84
RF	*physicochemical stats* (4)	50	0.76	0.75	0.76
RF	(1) and (3)	50	0.83	0.83	0.84
RF	(2) and (3)	100	0.84	0.84	0.83
MLP (L1)	(2) and (3)	-	0.83	0.89	0.78
MLP (L2-A)	L1 predictions	-	0.93	0.36	0.93
MLP (L2-B)	L1 predictions	-	0.93	0.34	0.94

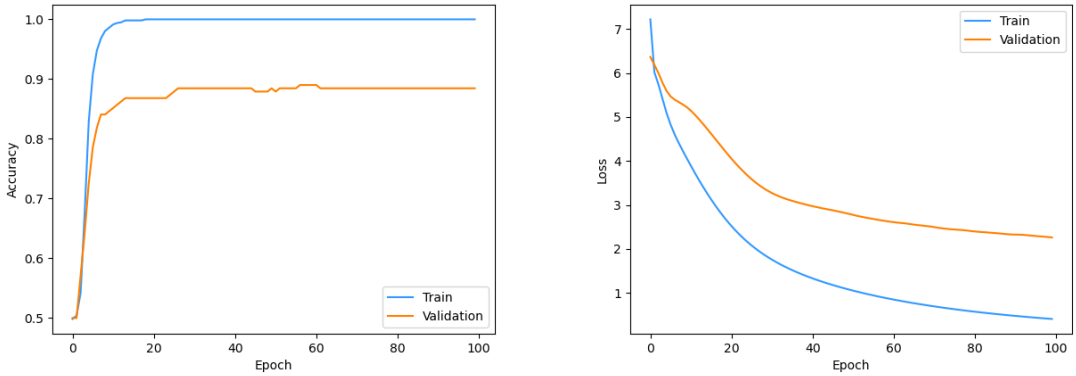

Fig. 4.1 L1 accuracy (left) and loss (right) for the training and validation sets at the end of each epoch for split 8 in the 10-fold cross-validation. The predicted class was the ones with the highest probability. The validation accuracy hits a plateau by epoch 70, despite the validation loss continuously decreases.

for the 9^{th} split in the 10-fold cross-validation. The accuracy plateaued at 90% in spite of the continuous decrease in validation loss. L1 performance metrics on the entire data are shown in Table 4.1. Its sensitivity increased with respect to the RF for the same feature set, but the specificity and accuracy decreased. For the PromClass second classification layer (L2) (Fig. 3.3-(b)) two models presented the highest AUC of 83%, L2-A and L2-B. Their performance over the test set is shown in Table 4.1. Accuracy of 93% for both and specificity of 93% and 94%, respectively. RF and L1 are applied on features extracted from promoters, L2 is applied on features extracted from the predictions made by L1.

We applied both L2 classifier variants of PromClass, CNNProm, and iPromoter-2L to a subset of the reference *E. coli* genome. The TP, FN and FP of the predictions made by the four solutions are shown in Table 4.2 and the location of positive predictions versus the RegulonDB promoters blasted to this genome segment locations are shown in Fig. 4.3. In all cases, the number of predicted promoters is much larger than the number of true promoters, reflecting the numbers in Table 4.2. We also calculated the precision and sensitivity for CNNProm and iPromoter-2L plotted in Fig. 4.2. This chart is the precision-recall (sensitivity) curve, but because we do not know how different thresholds would change Precision and Sensitivity for CNNProm and iPromoter-2L, they each have a single point in the chart. CNNProm presented 5.6% of precision with 45.2% for sensitivity whereas iPromoter-2L had 0.1% of precision with 100.0% for sensitivity. For PromClass, however, we calculated the pair (precision, sensitivity) using thresholds t $\in$ {90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, 10%}. The four values for PromClass in Table 4.2 are using thresholds that

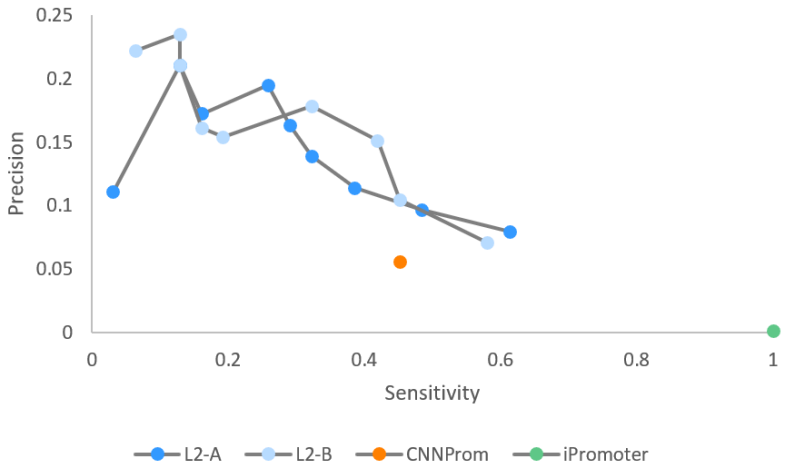

Fig. 4.2 Precision versus sensitivity for PromClass with L2-A (dark blue) and L2-B (light blue), CNNProm (orange) and iPromoter-2L (green). The points for L2-A and L2-B show the results of the models with threshold t $\in$ {90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, 10%}, from the left to the right.

Table 4.2 iPromoter-2L, CNNProm, L2-A and L2-B models' evaluation on the query-genome.

Model	TP	FN	FP
iPromoter-2L	31	0	22177
CNNProm	14	17	236
L2-A (threshold t=0.2)	15	16	140
L2-A (threshold t=0.1)	19	12	219
L2-B (threshold t=0.2)	14	17	120
L2-B (threshold t=0.1)	18	13	236

prompted higher or equal precision and sensitivity than CNNProm. On the contrary of PromClass or CNNProm that outputs a list of the positions where a promoter was predicted without overlaps, iPromoter-2L outputs the predictions made for all sequences in the segment overlapping each other by 80bp, i.e., there are several predictions for each region besides, promoters were predicted every where in the query as shown in Fig.4.3. In this image, one can observe there are so many green dots referring to iPromoter-2L positive predictions for promoters that they form a line. The 1456 positive predicted promoters overlapped the 31 true promoters in the query-genome; therefore, all true promoters were identified by iPromoter-2L and its TP = P = 31. The remaining 22177 positive predictions that did not overlap a true promoter were accounted as FP (Table 4.2). This approach penalizes itself for predicting the entire genome segment as promoter.

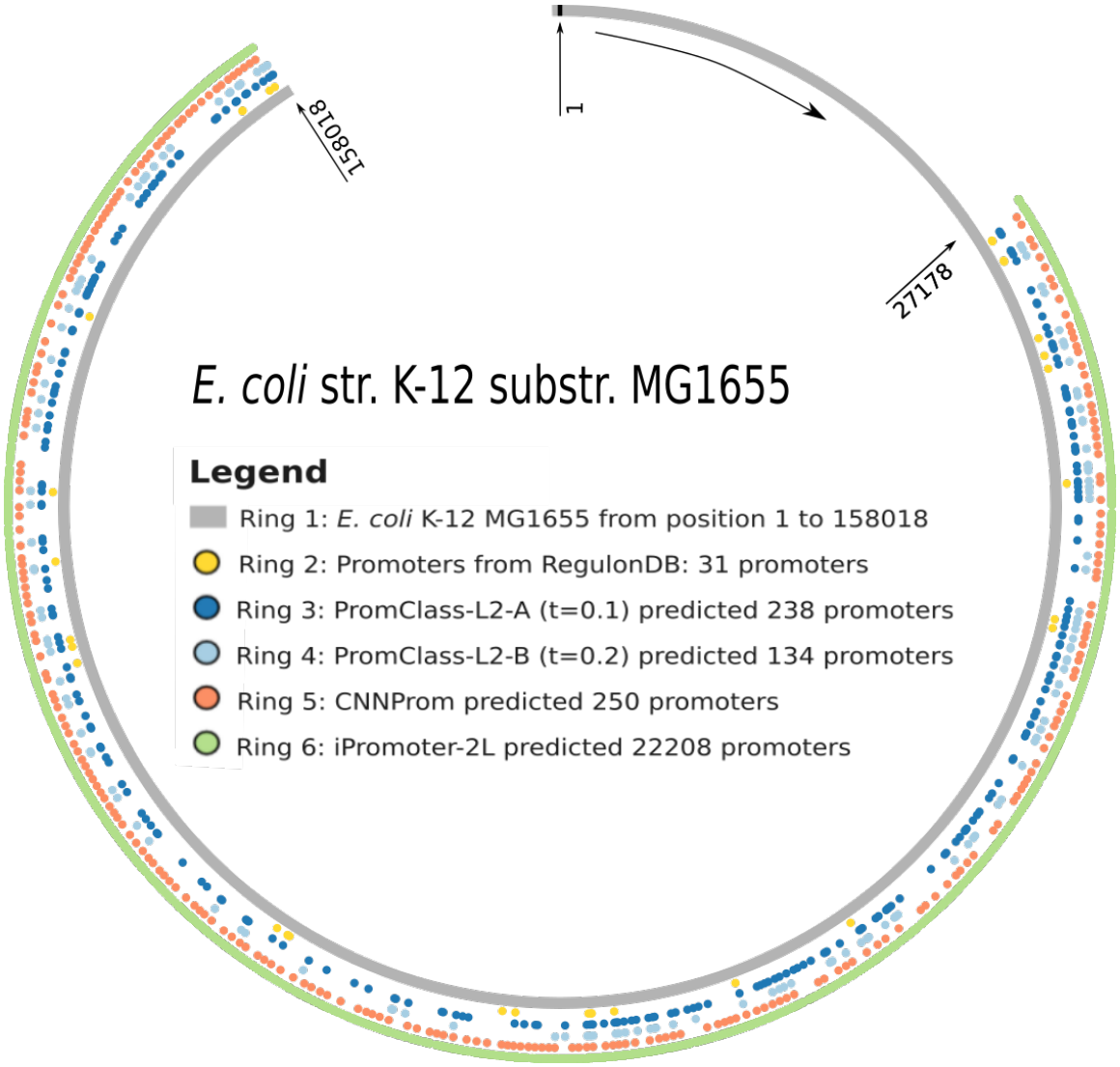

Fig. 4.3 Representation of *E.coli* genome from position 1 to 158018 with relative positions of RegulonDB promoters and the promoters predicted by, from the inside out, PromClass-L2-A t=0.1, PromClass-L2-B t=0.2, CNNProm, and iPromoter-2L shown for positions 27178 to 158018. This visualization is adapted from Overmars *et al.* (2015)'s tool.

4.2 Multiclass Multi-layer Perceptrons

The results for the multi-classes models are shown in the confusion matrix in Fig. 4.4.The seven-class model classifies the queries into non-promoter, σ24, σ28, σ32, σ38, σ54, or σ70. The confusion matrix shows that 86% of the non-promoter samples were correctly classified but this is the only percentage above 50%. For the remaining classes only 28%, 33%, 41%, 32%, 41% and 46% were correctly classified, respectively. In the confusion matrix, one can observe the misclassifications occur in favor of the classes with higher number of samples in the order non-promoter, σ70, σ38, σ24 and σ32.

One can also observe that, to some extent, when class A has samples misclassified as class B, class B will also have samples miss classified as class A. We call these classes common classes. Below are some examples:

- 6% = 4 samples from σ24 were miss classified as σ38 and 2% = 2 samples from σ38 were miss classified as σ24;
- 5% = 3 samples from σ24 were miss classified as σ32 and 5% = 3 samples from σ32 were miss classified as σ24;
- 10% = 105 samples from σ- - were miss classified as σ70 and 27% = 201 samples from σ70 were miss classified as σ- -;
- 39% = 40 samples from σ38 were miss classified as σ70 and 23% = 170 samples from σ70 were miss classified as σ38.

These bidirectional relationships might be due to similarity between samples from different classes sets. σ24, σ32, σ38 and σ70 are common classes. On the contrary, despite σ28's and σ54's samples were misclassified as belonging to other classes, the only samples classified into σ28 and σ54 are their own samples. They are referred as uncommon. These same two classes are also the classes with the fewest number of samples, 9 (or 0.4% of entire data set) and 17 (or 0.8% of entire data set), respectively, which may have caused these classes to not be a common label among the predictions. σ32 is not a common neither a uncommon class, but it fits in between. σ32 samples were misclassified as non-promoter, σ24, σ38 and σ70, but only samples from σ24 and σ70 were misclassified as σ32. This case will be referred to as less common. σ32 has also the third-smallest number of samples, 58 or 2% of this data set.

For the six-class model we discard the samples that were not promoters. This increased the proportion of all promoter classes in the training set. In its confusion matrix in Fig. 4.4,

the only uncommon case is σ28. Even with the increase of its representation in the data set, it is still low at 0.8%. σ54 that was uncommon in the seven-class model, now is only less-common. Its representation in the data set jumped to 1.7%. The other less-common class is σ24 that is the fifth smallest data set, but still it represents 6% of entire set and may be the reason why only 2% (1 sample) of σ24 samples were classified as σ54. Different from former behavior, σ54 now has its samples split between itself (53%) and σ70 (47%). The absence of the non-promoter group causes the classifier to learn more about σ54 and correctly classify most of its samples. Similarly, 55% of σ24, 71% of σ32, and 80% of σ70 were now correctly classified. The only two classes left behind were σ28 and σ38. As mentioned above, σ28 has only 0.8% of all samples and may be under-fitted and dominated by σ70 that has most of the samples. σ38, on the contrary, has the second-biggest set of samples and still most of them were misclassified as σ70. This relationship between σ38 and σ70 is bidirectional like in the seven-class model, i.e., samples from σ70 were also classified as σ38. In this model, however, the number of samples from σ70 misclassified as σ38 is almost twice smaller than before but still bigger than the number of samples from σ38 classified as σ70. Therefore, the other classes' samples being classified as σ70 may be due to σ70 being the dominant class, but the relationship between σ38 and σ70 suggests there are similarities between them. According to Mitchell *et al.* (2019), the protein σ-factors recruited by the RNAP to identify and bind the promoters σ38 and σ70 are very similar. The amino-acid sequence of the protein σ38 has 3 of the 4 domains σ70 amino-acid sequence is divided into. Mitchell *et al.* (2019) also says that based on sequence similarity, the protein σ-factors can be grouped into two families, the σ54 family and the σ70 family. All σ-factors belong to σ70 family, except σ54 that has its own family.

Conclusion and future work

5.1 Conclusion

Three of the RF models yielded a similar accuracy (83%) to iPromoter-2L (82%). The RF models used the features set (1) *physicochemical per 3-mers* alone, (2) combined with *pseknc* and (3) combined with *several k-mers*. Our analysis showed the sets of features

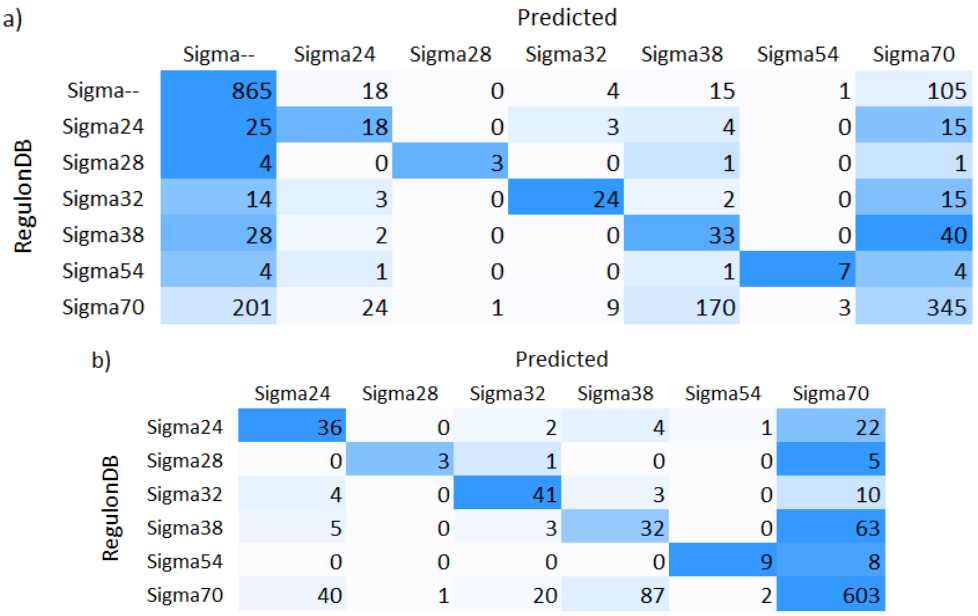

Fig. 4.4 Confusion matrix for the multi-class models. In (a) is the seven-class model and in (b) is the six-class model. In the row are the truth labels and in the columns are the predictions. The highest a number is with respect to its row, the darker the blue its cell was colored with.

including (*pseknc* and *several k-mers*) did not contribute to the RF classification process reaching similar scores than the model using *physicochemical per 3-mers* only. These models predicted too many false-positives promoters which motivated the use of a different classifier.

PromClass using MLP was developed to deal with more complex hidden patterns. Our approach has two layers of classification with preprocessing pipelines to create the features to feed the two classifiers. The first preprocessing step starts by extracting segments with 58 nucleotides from the genome and encoding them using *several k-mers* and *physicochemical properties over 3-mers*. Despite observing RF performing as good with *physicochemical properties over 3-mers* alone as when using it paired with *several k-mers*, we decided to exploit both feature sets (*physicochemical properties over 3-mers* and *several k-mers*) with the MLP model. *several k-mers* cannot be directly deduced from the *physicochemical properties over 3-mers* and, therefore, it may bring new knowledge to be discovered when explored by MLP instead of RF. The features are fed to PromClass-L1 that outputs the probability of the fed feature being a promoter. The second preprocessing step takes place and uses each of the 58 consecutive probabilities and their reverse complement probabilities to create feature vectors that are used as input for PromClass-L2. PromClass-L2 outputs the probability p of each nucleotide being in a promoter. The filter reads the nucleotides with p > threshold ρ and

determines, based on their relative position, if there is a promoter where those nucleotides are, or not. If yes, the promoter location is yielded.

PromClass-L1 is trained and tested with RegulonDB promoter samples and can be directly compared with iPromoter-2L and CNNProm. Table 4.1 shows L1's accuracy of 83% on the test set surpassing iPromoter-2L's accuracy of 82%. CNNProm's accuracy however is at 93% based on the numbers they reported for sensitivity and specificity and overcomes PromClass-L1 on the test set. PromClass-L1 gives intermediary results in the real-life problem of finding promoters on a stream of DNA nucleotides. PromClass-L2-A and L2-B were applied to positions 27178 to 158018 in the genome using cross-validation and then submitted to a filter. The same Table 4.1 presents L2-A and L2-B scores on the test set. The specificity is high at 93% and 94%, respectively, showing that the models' FP predictions happen only 7 and 6% of the time, but the sensitivity is low what means most promoters are not being identified. However, considering that roughly 99% of the database is composed by samples of non-promoters, the easy result would be to predict all the samples as non-promoters and that would be correct 99% of the time with zero sensitivity for promoters. Our models were able to bring this sensitivity up to 36 and 34% keeping the FP rate low as intended.

Neither iPromoter-2L or CNNProm reported scores that were suitable for classifying sequences extracted from a genome. Therefore, we applied iPromoter-2L and CNNProm to the same DNA positions 27178 to 158018. iPromoter-2L predicted 22,177 promoters sequences but only 0.14% of the promoter predictions are true promoters (Table 4.2). It is important to remember these results are for a query that is only 3% of the full *E. coli* genome. Despite iPromoter-2L's high sensitivity, the high number of FP make this a non-practical solution for classification at the whole-genome level.

On the other hand, Umarov and Solovyev (2017)'s solution makes far fewer FP predictions reaching 5.6% of precision with 236 FP. However, despite reporting sensitivity and specificity on *E. coli*'s test set at 90% and 96%, respectively, CNNProm found only 14 of the 31 promoters in the query which gives a sensitivity of only 45%. These true promoters are samples from RegulonDB and were used on CNNProm as well as on iPromoter-2L and PromClass training. It emphasizes the great difficulty of identifying promoters at genomic scales as opposed to classifying fixed length sequences either with a complete promoter in it or completely free of promoter pieces. It also raises the question of how representative are the negative samples used by CNNProm. CNNProm's authors create their negative set by using the reverse complement of randomly selected genes and were able to reach the claimed high sensitivity and specificity. In comparison, with the same number of FP made by CNNProm, PromClass-L2-B threshold t=10% has higher sensitivity, 58% against 45% from CNNProm, and higher precision, 7% against 5.6%. Similarly, with the same sensitivity

presented for CNNProm, PromClass-L2-B threshold t=20% has a lot less FP: precision of 10% against 5.6% from CNNProm. Besides, PromClass-L2-A threshold of t=10% presents higher sensitivity and precision than CNNProm, 61% and 8%, respectively. These outcomes are shown in the Precision vs Sensitivity plot in Fig. 4.2. These are reiterated examples of PromClass-L2s overcoming CNNProm despite the low sensitivity reported in Table 4.1 for the first ones and the high sensitivity reported by Umarov and Solovyev (2017) for the last one.

Finally, when classifying the promoters into σ-factors or into non-promoters and σ-factors, we observed much worse classification scores. The low scores may be due to too few samples as for σ28, to high similarity between classes such as σ38 and σ70, and a challenging set of negative samples that is very similar to promoter samples.

5.2 Future work

5.2.1 Reducing prediction time

The pre-processing steps make the approach very time-consuming. We observed three main causes. First is the use of a sliding window over the genome's forward and reverse strands. We chose a sliding window of 58 nucleotides while iPromoter-2L and CNNProm both use a window of 81 nucleotides. However, for a genome with 4 million nucleotides, both windows sizes will generate over 4 million promoter's candidate sequences for each DNA strand. This is too many sequences to be processed. The second one is with respect to the encoding algorithms we chose to encode each candidate, *physicochemical per 3-mers* and *several k-mers*. *physicochemical per 3-mers* requires the sequence to be decomposed into 3-mers that is then looked up on a table. However, *several k-mers* decomposes the sequence into k-mers for k values from 1 to 6. The entire *several k-mers*'s vector has 5460 dimensions which 4096 dimensions refer only to 6-mer and is sparse. Calculating and manipulating 6-mer is timing and memory consuming yet most of its data are zeros. For future work we will put more effort on performing feature selection. The third and last one is to perform a strategic re-implementation of these approaches, recycling former calculated steps and using the GPU as much as possible.

5.2.2 Using *E. coli* to predict promoter sequences in other species

The seven σ factors 19, 24, 28, 32, 38, 54 and 70 present in *E. coli* are also found in other organisms. σ70, for instance, can be found also in *Bacteroides fragilis* (Mastropaolo *et al.* (2009)), *Streptomyces coelicolor*, *Caulobacter cresentus* and *Mycobacterium tuberculosis*

(Paget and Helmann (2003)). σ24, 28, 38, 54 and 70 are found in *S. enterica* (Maloy and Edwards (1999)) as well as σ32 (Testerman *et al.* (2002)). According to McClelland *et al.* (2001), there is 80% of similarity between *E. coli* and *S. enterica* coding regions. Therefore, our hypothesis is that the promoters from these organisms must present some conservation across the different species to be recognized by sigma factors with the similar characteristics that as in *E.coli*, specially if the coding regions that encode sigma factors in these species and in *E.coli* are similar. These similarities between *E. coli* and other species suggest that a model trained to identify promoters on *E. coli* may succeed on identifying promoters on other species. Therefore, our future work is to understand these relationships of similarities, apply PromClass to the DNA of other species and find a way to measure its performance to validate it.

References

Alipanahi, B., Delong, A., Weirauch, M.T. and Frey, B.J. (2015) (DEEPBIND)Predicting the sequence specificities of Dn A- and Rn A-binding proteins by deep learning *nature biotechnology* **33**(8), pp. 3–1

Beiko, R.G. (2003) Evolutionary computing strategies for the detection of conserved patterns in genomic dna Ph.D. Thesis University of Ottawa (Canada)

Beiko, R.G. and Charlebois, R.L. (2005) Gann: genetic algorithm neural networks for the detection of conserved combinations of features in dna *BMC bioinformatics* **6**(1), p. 36

Chen, W., Lei, T.Y., Jin, D.C., Lin, H. and Chou, K.C. (2014) PseKNC: A flexible web server for generating pseudo K-tuple nucleotide composition *Analytical Biochemistry* **456**, pp. 53–60

Chen, W., Zhang, X., Brooker, J., Lin, H., Zhang, L. and Chou, K.C. (2015) PseKNC-General: a cross-platform package for generating various modes of pseudo nucleotide compositions *Bioinformatics* **31**(1), pp. 119–120

Chollet, F. *et al.* (2015) Keras https://keras.io

Dao, F.Y., Lv, H., Wang, F., Feng, C.Q., Ding, H., Chen, W. and Lin, H. (2019) Identify origin of replication in Saccharomyces cerevisiae using two-step feature selection technique *Bioinformatics* **35**(12), pp. 2075–2083

Davis, M.C., Kesthely, C.A., Franklin, E.A. and MacLellan, S.R. (2016) The essential activities of the bacterial sigma factor *http://dx.doi.org/10.1139/cjm-2016-0576* **63**(2), pp. 89–99

Demeler, B. and Zhou, G. (1991) Neural network optimization for e. coli promoter prediction *Nucleic acids research* **19**(7), pp. 1593–1599

Florquin, K., Saeys, Y., Degroeve, S., Rouze, P. and Van de Peer, Y. (2005) Large-scale structural analysis of the core promoter in mammalian and plant genomes *Nucleic acids research* **33**(13), pp. 4255–4264

Gama-Castro, S., Salgado, H., Santos-Zavaleta, A., Ledezma-Tejeida, D., Muñiz-Rascado, L., García-Sotelo, J.S., Alquicira-Hernández, K., Martínez-Flores, I., Pannier, L., Castro-Mondragón, J.A., Medina-Rivera, A., Solano-Lira, H., Bonavides-Martínez, C., Pérez-Rueda, E., Alquicira-Hernández, S., Porrón-Sotelo, L., López-Fuentes, A., Hernández-Koutoucheva, A., Moral-Chávez, V.D., Rinaldi, F. and Collado-Vides, J. (2016) RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond *Nucleic Acids Research* **44**(D1), pp. D133–D143

Gordon, L., Chervonenkis, A.Y., Gammerman, A.J., Shahmuradov, I.A. and Solovyev, V.V. (2003) Sequence alignment kernel for recognition of promoter regions *Bioinformatics* **19**(15), pp. 1964–1971

Hohmann-Marriott, M. and Lale, R. (2020) MODULATION OF GENE EXPRESSION - Norwegian University of Science and Technology (NTNU)

Hooghe, B., Broos, S., van Roy, F. and De Bleser, P. (2012) A flexible integrative approach based on random forest improves prediction of transcription factor binding sites. *Nucleic acids research* **40**(14), p. e106

Kornberg, R.D. (1999) Eukaryotic transcriptional control *Trends in Biochemical Sciences* **24**(12), pp. 46–49

Lai, H.Y., Zhang, Z.Y., Su, Z.D., Su, W., Ding, H., Chen, W. and Lin, H. (2019) iProEP: A Computational Predictor for Predicting Promoter *Molecular Therapy - Nucleic Acids* **17**, pp. 337–346

Lin, H., Deng, E.Z., Ding, H., Chen, W. and Chou, K.C. (2014) iPro54-PseKNC: a sequence-based predictor for identifying sigma-54 promoters in prokaryote with pseudo k-tuple nucleotide composition *Nucleic Acids Research* **42**(21), pp. 12961–12972

Lin, H. and Li, Q.Z. (2011) Eukaryotic and prokaryotic promoter prediction using hybrid approach *Theory in Biosciences* **130**(2), pp. 91–100

Lin, H., Liang, Z.Y.Y., Tang, H. and Chen, W. (2017) Identifying sigma70 promoters with novel pseudo nucleotide composition *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **16**(4), pp. 1–1

Lin, M. and Guo, J.t. (2019) New insights into protein–DNA binding specificity from hydrogen bond based comparative study *Nucleic Acids Research* **47**(21), pp. 11103–11113

Liu, B., Yang, F., Huang, D.S. and Chou, K.C. (2018) iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC *Bioinformatics* **34**(1), pp. 33–40

Mallios, R.R., Ojcius, D.M. and Ardell, D.H. (2009) An iterative strategy combining biophysical criteria and duration hidden markov models for structural predictions of chlamydia trachomatis σ 66 promoters *BMC bioinformatics* **10**(1), p. 271

Maloy, S. and Edwards, R. (1999) salmonella.org

Mastropaolo, M.D., Thorson, M.L. and Stevens, A.M. (2009) Comparison of Bacteroides thetaiotaomicron and Escherichia coli 16S rRNA gene expression signals *Microbiology* **155**(8), pp. 2683–2693

McClelland, M., Sanderson, K., Spieth, J., Clifton, S., Latreille, P., Courtney, L., Porwollik, S., Ali, J., Dante, M., Du, F. and Hou, S. (2001) Complete genome sequence of *Salmonella enterica serovar* Typhimurium LT2

Min, S., Lee, B. and Yoon, S. (2016) Deep learning in bioinformatics *Briefings in Bioinformatics* (June), p. bbw068

Mitchell, A.L., Attwood, T.K., Babbitt, P.C., Blum, M., Bork, P., Bridge, A., Brown, S.D., Chang, H.Y., El-Gebali, S., Fraser, M.I., Gough, J., Haft, D.R., Huang, H., Letunic, I., Lopez, R., Luciani, A., Madeira, F., Marchler-Bauer, A., Mi, H., Natale, D.A., Necci, M., Nuka, G., Orengo, C., Pandurangan, A.P., Paysan-Lafosse, T., Pesseat, S., Potter, S.C., Qureshi, M.A., Rawlings, N.D., Redaschi, N., Richardson, L.J., Rivoire, C., Salazar, G.A., Sangrador-Vegas, A., Sigrist, C.J., Sillitoe, I., Sutton, G.G., Thanki, N., Thomas, P.D., Tosatto, S.C., Yong, S.Y. and Finn, R.D. (2019) InterPro in 2019: improving coverage, classification and access to protein sequence annotations *Nucleic Acids Research* **47**(D1), pp. D351–D360

Overmars, L., van Hijum, S.A.F.T., Siezen, R.J. and Francke, C. (2015) CiVi: circular genome visualization with unique features to analyze sequence elements: Fig. 1. *Bioinformatics* **31**(17), pp. 2867–2869

Paget, M.S. (2015) Bacterial sigma factors and anti-sigma factors: Structure, function and distribution *Biomolecules* **5**(3), pp. 1245–1265

Paget, M.S. and Helmann, J.D. (2003) The sigma70 family of sigma factors. *Genome Biology* **4**(1), p. 203

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011) Scikit-learn: Machine learning in Python *Journal of Machine Learning Research* **12**, pp. 2825–2830

Rahman, M.S., Aktar, U., Jani, M.R., Shatabda, S., Siddiqur Rahman, M., Aktar, U., Rafsan Jani, M. and Shatabda, S. (2019) iPro70-FMWin: identifying Sigma70 promoters using multiple windowing and minimal features *Molecular Genetics and Genomics* **294**(1), pp. 69–84

Saito, T. and Rehmsmeier, M. (2015) The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets *PLOS ONE* **10**(3), p. e0118432

Shafee, T. and Lowe, R. (2017) Eukaryotic and prokaryotic gene structure *WikiJournal of Medicine* **4**(1), pp. 0–3

Silva, S.d.A.e. and Echeverrigaray, S. (2012) Bacterial Promoter Features Description and Their Application on E. coli in silico Prediction and Recognition Approaches in: *Bioinformatics*, (Ed.) H. Pérez-Sánchez chap. 10 InTechOpen available at https://doi.org/10.5772/48149

Song, K. (2012) Recognition of prokaryotic promoters based on a novel variable-window z-curve method *Nucleic acids research* **40**(3), pp. 963–971

Tahir, M., Hayat, M., Gul, S. and Chong, K.T. (2020) (2L-iPSW(word2vec)) An intelligent computational model for prediction of promoters and their strength via natural language processing *Chemometrics and Intelligent Laboratory Systems* **202**, p. 104034

Tayara, H., Tahir, M. and Chong, K.T. (2020) Identification of prokaryotic promoters and their strength by integrating heterogeneous features *Genomics* **112**(2), pp. 1396–1403

Testerman, T.L., Vazquez-Torres, A., Xu, Y., Jones-Carson, J., Libby, S.J. and Fang, F.C. (2002) The alternative sigma factor σ^E controls antioxidant defences required for *Salmonella* virulence and stationary-phase survival *Molecular Microbiology* **43**(3), pp. 771–782

Umarov, R.K. and Solovyev, V.V. (2017) Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks *PLOS ONE* **12**(2), p. e0171410

Weiss, V., Medina-Rivera, A., Huerta, A.M., Santos-Zavaleta, A., Salgado, H., Morett, E. and Collado-Vides, J. (2013) Evidence classification of high-throughput protocols and confidence integration in RegulonDB *Database* **2013**, p. bas059

Xiao, X., Xu, Z.C., Qiu, W.R., Wang, P., Ge, H.T. and Chou, K.C. (2018) iPSW(2L)-PseKNC: A two-layer predictor for identifying promoters and their strength by hybrid features via pseudo K-tuple nucleotide composition *Genomics* **111**(6), pp. 1785–1793

Zuo, Y.C. and Li, Q.z. (2010) The hidden physical codes for modulating the prokaryotic transcription initiation *Physica A: Statistical Mechanics and its Applications* **389**(19), pp. 4217–4223